Welcome to the Sixth CBK Domain:

In this domain we cover:

- Assessment and test strategies.
  - How and what do want to test? Which type of tests do we use, and do we want intrusive or non-intrusive?
- Security process data (e.g., management and operational controls).
  - Are our administrative processes as secure as we think they are, and as they should be?
- Security control testing.
  - We test both the technical and administrative controls we have in place.
- Test outputs (e.g., automated, manual).
  - How we report our findings, we need to do this as effectively as possible.
  - We in layman's terms convey what the vulnerabilities and test results mean to senior management.
- Security architecture vulnerabilities.

This chapter focuses on how we assess and test the security measures we have in place, this is done to ensure we are as secure as we think we are and to improve our security posture.
CBK 6 makes up 12% of the exam questions.

Assessment and Test Strategies:

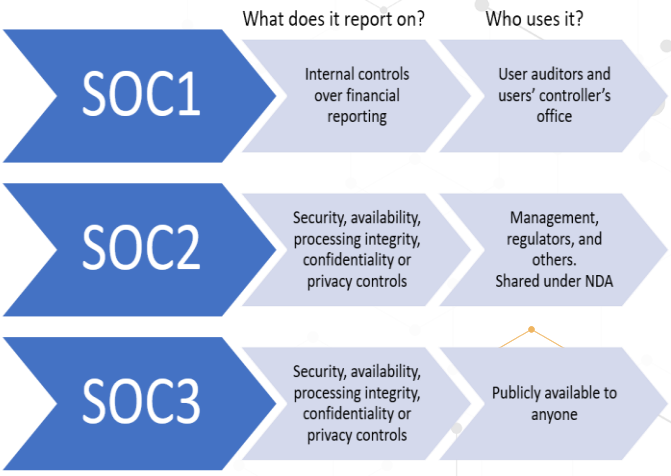
- **Key Terms:**
  - **Static Testing** – We passively test the code, we do not run it.
  - **Dynamic Testing** – We tests code while executing it.
  - **Fuzzing (Fuzz Testing)** – A black box testing that submits random, malformed data as inputs into software programs to determine if they will crash.
  - **Penetration Testing (Pen Testing)** – We pay someone to test our security by trying to compromise our safeguards. This is testing both our organization's physical and logical perimeter.
  - **Synthetic Transactions/Monitoring** - Building scripts or tools that simulate normal user activity in an application.
- **Security Assessments:**
  - A full picture approach to assessing how effective our access controls are, they have a very broad scope.
  - Security assessments often span multiple areas, and can use some or all of these components:
    - Policies, procedures, and other administrative controls.
    - Assessing the real world-effectiveness of administrative controls.
    - Change management.
    - Architectural review.
    - Penetration tests.
    - Vulnerability assessments.
    - Security audits.

- **Security Audit:** A test against a published standard.
  - SOC 2 Type 1 or 2, PCI-DSS, HIPAA…
- **Internal, external, and 3rd-Party Audits:**
  - **Unstructured Audits (internal):**
    - Internal auditors to improve our security and find flaws, often done before an external audit.
  - **External audits:**
    - Similar to internal audits. An external company audits our controls to find flaws and improve our security posture.
  - **Structured Audits (3rd party):**
    - External auditors who validate our compliance, often done for a regulatory body, they are experts and the audit adds credibility.
    - Can also be a knowledge transfer for the organization, required annually in many organizations.

- **SOC1:** Focus on service organization controls relevant to internal control over **financial reporting**.
  - For internal use and available to the organization.
  - **Type I:** Opinion on **design effectiveness** of controls. Type I, cover single point in time.
  - **Type II:** Opinion on **design and reporting effectiveness** of controls. Type II, covers a minimum six month time period.

- **SOC 2:** Assess internal controls for **compliance and operations.**
  - Must meet **trust service criteria** defined by AICPA: **Security, Availability, Processing Integrity, Confidentiality,** and **Privacy.**
  - **Type I:** Report on management's description of a service organization's system and the **suitability** of the design of controls. Type I, cover single point in time.
  - **Type II:** Report on management's description of a service organization's system and the **suitability** of the design and operating **effectiveness** of controls. Type II, covers a minimum six month time period.
    - The purpose of type II reports is to validate/verify that an organization meets the requirements as stated in the published standard.
    - Proves organization controls listed are operational and enforced.
    - Better and more expensive than Type I reports.

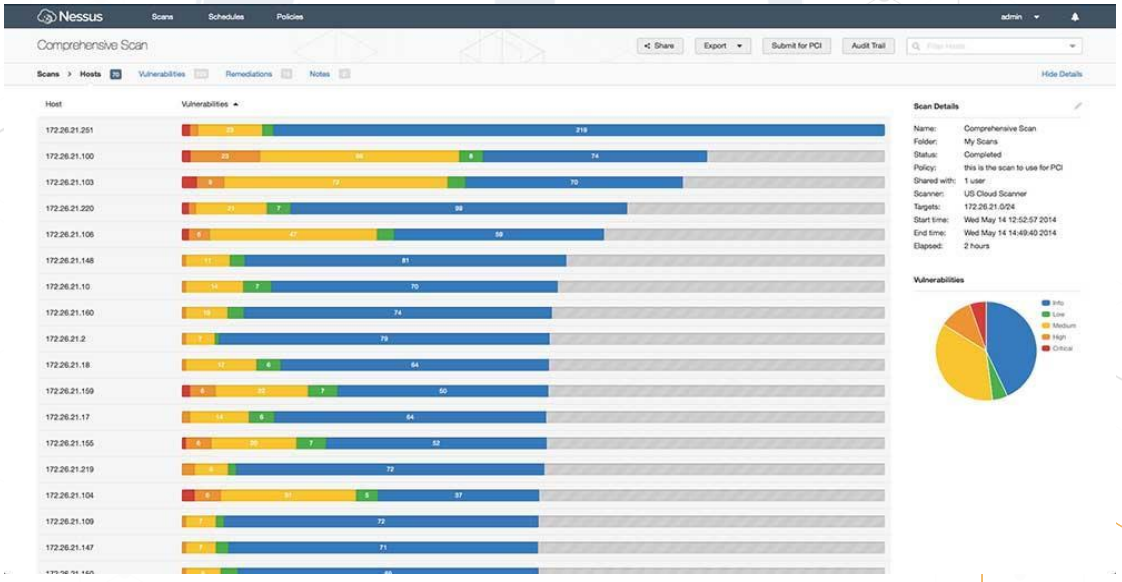
- **SOC 3:** Similar to SOC 2 report but much more generalized, shorter, and a less sensitive document.
  - More public facing document
  - Includes only the auditor's opinion and limited description of controls
  - The examination covers both design and operating effectiveness of the controls relevant to applicable trust service principles (security availability, processing integrity , confidentiality, and privacy).



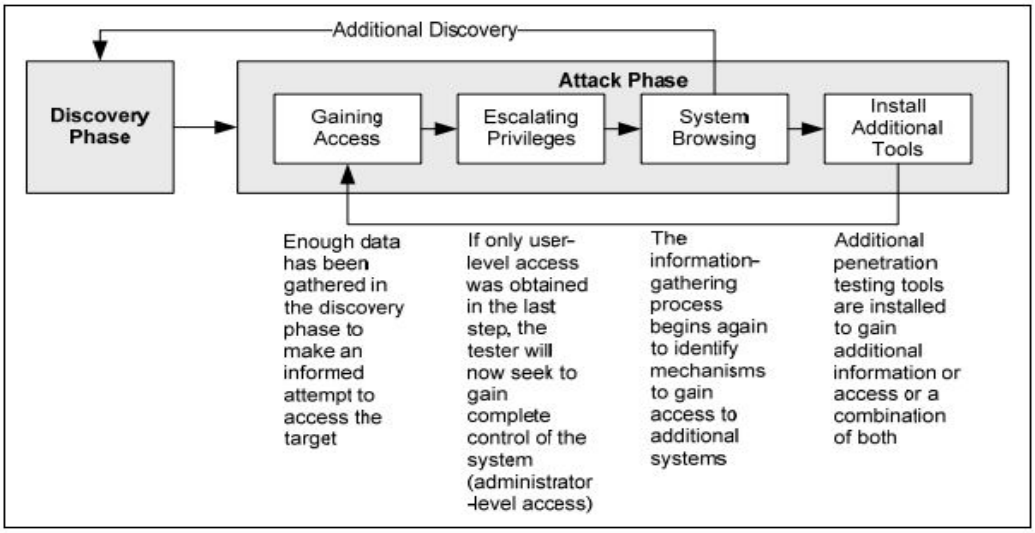
- **Security Audit Logs:**
  - Reviewing security audit logs in an IT system is one of the easiest ways to verify that access control mechanisms are working as intended.
  - Reviewing audit logs is primarily a detective control.
  - NIST Special Publication 800-92 suggests the following log types should be collected and audited:
    - **Network Security Software/Hardware:**
      - Antivirus logs, IDS/IPS logs, remote access software (such as VPN logs), web proxy, vulnerability management, authentication servers, routers and firewalls.
    - **Operating System:**
      - System events, audit records, applications, client requests and server responses, usage information, significant operational actions.
  - **Centralized Logging:**
    - Should be automated, secure and even administrators should have limited access.
    - Often a central repository is hashed and never touched, and a secondary copy is analyzed to ensure integrity.
    - Logs should have a retention policy to ensure we are compliant and we keep the logs as long as we need them.
    - Checking logs is often an afterthought and rarely done, where do we start?
    - Since they are often keeping everything, there can be 10's of millions of lines of log info, we need to implement systems to automate this as much as makes sense.

- **Security Audit Logs (Audit Trail):**
  - Audit record management typically faces five distinct problems:
    1. Logs are not reviewed on a regular and timely basis.
    2. Audit logs and audit trails are not stored for a long enough time period.
    3. Logs are not standardized or viewable by correlation toolsets - they are only viewable from the system being audited.
    4. Log entries and alerts are not prioritized.
    5. Audit records are only reviewed for the bad stuff.

- **Vulnerability Scanning/Testing:**
  - A vulnerability scanner tool is used to scan a network or system for a list of predefined vulnerabilities such as system misconfiguration, outdated software, or a lack of patching.
  - It is very important to understand the output from a vulnerability scan, they can be 100's of pages for some systems, and how do the vulnerabilities map to Threats and Risks (Risk = Threat x Vulnerability).
  - When we understand the true Risk, we can then plan our mitigation.
  - Common vulnerability scanners could be Nessus or OpenVAS, both list vulnerabilities in Critical, High, Medium, Low, and Informational.



- **Penetration Testing (Pen Testing),** often called Ethical Hacking:
  - Test if the vulnerabilities are exploitable.
  - An authorized simulated attack on our organization that looks for security weaknesses, potentially gaining access to the systems, buildings and data.
  - It is very important to have very clear rules of engagement defined in a SOW (Statement Of Work)
    - Which IP ranges, time frame, tools, POC, how to test, what to test, …
    - We confirm with our legal team before hiring Pen Testers, even if you allow it what they do may still be illegal.
  - Senior management set the goals for the Pen testing.
    - Why are we doing it? What are we trying to achieve? They have to sign off on it.
  - If we are the pen testers, we are there to test and document the vulnerabilities, not to fix them.
    - We provide the report to senior management and they decide which vulnerabilities they want to address.
  - Use multiple attack vectors and Pen testing uses an iterative process that is similar to Agile project planning.
  - **Discovery** (planning): Finding the vulnerabilities, design the attacks.
  - **Gaining Access**: Access the network.
  - **Escalate Privileges**: Get higher-level access, ultimately we want admin access.
  - **System Browsing**: Gain additional access, often back to discovery again with our new knowledge level and access.
  - **Install Additional tools**: With our elevated access we can install more tools and exploit new attack surfaces, can go back to Gaining Access.
  - Finally when done, they report the findings.



- Planning > Reconnaissance > Scanning (enumeration) > Vulnerability assessment > Exploitation > Reporting.
- Very similar to a black hat methodology.
- Black hats often spend less time planning, and instead of reporting they cover their tracks.
    - They delete/modify logs and any other tracks they left and if possible install backdoors, so they can keep exploiting our environments.
- A Pen tester has a very clear SOW and they do not compromise system and data integrity.
- The Pen tester may also not be allowed to access certain files (PII/PHI), but a dummy file is created in the same location, if the Pen tester can get to the target file, they could get to the actual data file.
- The Pen testing is done in clearly defined time windows, often in maintenance windows after hours, the point is to prove we are vulnerable, not disrupt our business.
- Some low impact Pen tests can also be done on DR environments, to not effect our live environments, but they are often less useful since most DR environments are not a mirror copy of the production environment.
- **Black Box Pen Testing (Zero Knowledge):**
    - The attacker had no knowledge about the organization other than publicly available information.
    - They start from the point an external attacker would.
- **White (Crystal/Clear) Box Pen testing (Full Knowledge):**
    - The attacker has knowledge of the internal network and access to it like a privileged employee would.
    - Normally Administrator access employee with full knowledge of our environment.
- **Gray (Grey) Box Pen Testing (Partial Knowledge):**
    - The attacker has limited knowledge, a normal user, vendor or someone with limited environment knowledge.
- Do not confuse these with Black, Gray, or White Hat Hackers.
    - **White Hat Hackers:** Professional Pen Testers trying to find flaws so we can fix it (Ethical Hackers).
    - **Black Hat Hackers:** Malicious hackers, trying to find flaws to exploit them (Crackers - they crack the code).
    - **Gray/Grey Hat Hackers:** They are somewhere between the white and black hats.
- **Breach and Attack Simulation (BAS):**
    - Uses automated tools to simulate complex cyberattacks on-demand across all attack vectors.
    - Combines red and blue team approaches (purple teaming) — automates it and provides us with breach and assault platforms with continuous coverage.
    - They can be running 24/7/365, giving us much more in-depth visibility into the real state of our defense readiness.

- Breach simulation is used to simulate attacks on endpoints (malware), data exfiltration, malware attacks, APT attacks that move laterally through a network, targeting the most valuable assets.

- **Compliance Checks:**
  - Are the security controls we put in place sufficient to ensure compliance with the regulations that our organization must follow? (PCI-DSS, HIPAA, SOC2, and so on).
  - Audits can be part of it, but they are a point-in-time event, whereas compliance checks are ongoing, and compliance should be the beginning of our risk management program.
- Think like an attacker would, start with the easiest attack first, the users.
- Low technical tools can be just as effective as sophisticated tools,
- Many organizations have strong perimeter defense, but no defense in depth, once you get past 1 or 2 barriers you can access most things.
- **Social Engineering** uses people skills to bypass security controls.
  - Can be used in a combination with many other attacks, especially client-side attacks or physical tests.
  - Attacks are often more successful if they use one or more of these approaches:
    - **Authority (someone you trust or are afraid of)** - Look and sound like an authority figure, be in charge, this can be in a uniform or a suit. Most effective with impersonation, whaling, and vishing attacks.
    - **Intimidation (If you don't bad thing happens)** - Virus on the network, credit card compromised, lawsuit against your company, intimidation is most effective with impersonation and vishing attacks.

- **Social Engineering Attacks:**
  - **Consensus (Following the crowd, everyone else was doing it)** - Fake reviews on a website, using consensus/social proof is most effective with Trojans and hoaxes.
  - **Scarcity (If you don't act now, it is too late)** - New iPhone out, only 200 available, often effective with phishing and Trojan attacks.
  - **Urgency (It has to happen now or else)** - The company will be sued for $1,000,000 if these papers are not filled out before Friday, often used with Phishing.
  - **Familiarity (Have a common ground, or build it)** - Knowing something about the victim ahead of time and then reference it can raises chances of a successful attack drastically. People want to be helpful, if they feel like they know you they want to even more. Often successful with vishing and in-person social engineering.

- **War Dialing:**
  - Uses modem to dial a series of phone numbers, looking for an answering modem carrier tone, the penetration tester then attempts to access the answering system.
  - Not really done anymore, but know it for the exam.
- **War Driving** (access point mapping):
  - Driving or walking around, mapping access points and trying to gain access to them.
- **Network Attacks**
  - Client-side attacks, server-side attacks, or Web application attacks.
- **Wireless Tests:**
  - Evaluate the risk related to potential access to your wireless network.
  - Uses the password combination & sniffing technique for cracking unsecured wireless network, so a proper set up is required for making the whole process semi-automated and automated.

- **Penetration Testing Tools and Methodology:**
  - Just like hackers, Pen testers use many different tools to test, both published tools and own creations.
  - Be VERY careful if testing these out, do not use them outside your own network, and only on internal networks with written permission.
  - Penetration testing tools:
    - Open source Metasploit - http://www.metasploit.org/
    - Closed source Core Impact - http://www.coresecurity.com/
    - Immunity Canvas - http://www.immunitysec.com/
    - Top 125 Network Security Tools - http://sectools.org/
    - Kali Linux - https://www.kali.org/
- **Semi Real-Time Attack Maps:**
  - https://cybermap.kaspersky.com/
  - https://www.digitalattackmap.com/

- **Exception Handling:**
  - An exception is raised or thrown when an application encounters an error (programming error, division by zero, invalid argument, creating an object when the system is out of memory, and so on).
  - Most applications would by default terminate, but an exception handler can stop that.

- **Ethical Disclosure:**
  - As IT Security professionals, we need to act ethically — disclose unknown vulnerabilities discovered during security testing.
  - We'd most likely put in place compensating controls to address the vulnerability.
  - We'd notify the vendor, giving them time to create a patch or other form of fix (white hat).

- If they do not act, we may disclose it to a larger audience; however, this raises security concerns now that attackers are aware of the vulnerability — they can attack before others have applied to compensate controls or repair.

- **Software Testing:**
  - Historically we have built functional software and tested it for just that stability and functionality, security has been an afterthought if considered at all. Software needs to be designed securely, built in not bolted on.
  - Normal software can have millions of line of code and about 1% of that contains vulnerabilities.
  - Many security breaches happen because our software is easy to compromise.
  - **Static Testing -** Passively testing the code, it is not running.
    - This is walkthroughs, syntax checking, and code reviews.
    - Looks at the raw source code itself looking for evidence of known insecure practices, functions, libraries, or other characteristics having been used in the source code.
    - There are 100's of static code analysis tools available depending on programming language.
  - **Dynamic Testing** – Actively testing the code while executing it.
    - Can uncover flaws that exist in the particular implementation and interaction of code that static analysis missed. Software can run and code execute with flaws.
  - Code testing uses white and black box terms just like in Pen testing.
    - **White Box Software Testing:**
      - The tester has full access to program source code, data structures, variables,...
    - **Black Box Software Testing:**
      - The tester has no details, just the software, they then test for functionality and security flaws.
  - **TM/RTM (Requirements Traceability Matrix):**
    - Normally a table, used to map customer requirements to the testing plan using a many-to-many relationship comparison.
    - A requirements traceability matrix may be used to check if the current project requirements are being met, and to help in the creation of a request for proposal, software requirements specification, various deliverable documents, and project plan tasks.

Requirement Identifiers	Reqs Tested	REQ1 UC 1.1	REQ1 UC 1.2	REQ1 UC 1.3	REQ1 UC 2.1	REQ1 UC 2.2	…
Test Cases	321	3	2	3	1	1	
Tested Implicitly	77						
1.1.1	1	x					
1.1.2	2		x	x			
1.1.3	2	x					
1.1.4	1			x			
1.1.5	2	x					
…							

- **Software Testing Levels:**
  - **Unit Testing:**
    - Tests that verify the functionality of a specific section of code.
    - In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.
    - Usually written by developers as they work on code (white-box), to ensure that the specific function is working as expected.
  - **Integration Testing:**
    - Seeks to verify the interfaces between components against a software design.
    - Integration testing works to expose defects in the interfaces and interaction between integrated components/modules.
    - Progressively larger groups of software components are tested until the software works as a system.
  - **Component Interface Testing:**
    - Testing can be used to check the handling of data passed between various units, or subsystem components, beyond full integration testing between those units.
    - Tests a completely integrated system to verify that the system meets its requirements.
  - **Operational Acceptance:**
    - Used to conduct operational readiness (pre-release) of a product, service or system as part of a quality management system.

- **Software Testing Types:**
  - **Installation Testing:**
    - Assures that the system is installed correctly and working at actual customer's hardware.
  - **Regression Testing:**
    - Finding defects after a major code change has occurred.
    - Looks for software regressions, as degraded or lost features, including old bugs that have come back.
  - **Fuzzing (Fuzz Testing):**
    - Testing that provides a lot of different inputs in order to try to cause unauthorized access or for the application to enter unpredictable state or crash.
    - If the program crashes or hangs the fuzz test failed.
    - The Fuzz tester can enter values into the script or use pre-compiled random or specific values.
    - **Mutating fuzzing** – The tester analyses real info and modifies it iteratively.

- **All-Pairs Testing (Pairwise Testing):**
  - All-Pairs Testing is defined as a black-box test design technique in which test cases are designed to execute all possible discrete combinations of each pair of input parameters.
  - The most common bugs in a program are generally triggered by either a single input parameter or an interaction between pairs of parameters.
  - It uses carefully chosen test vectors, this can be done much faster than an exhaustive search of all combinations of all parameters, by parallelizing the tests of parameter pairs.
  - If we have a very simple piece of software with 3 input parameters:
  - Server type A: Physical B: VM Vendor: A: Dell B: HP
    Serial number: A Valid (5000) B Invalid
    - → If we test all possible combinations we would test 2x2x5000 = 20000 combinations.
    - → If we test all-pairs we would test 2x2x2 = 8 combinations – we only look at valid or invalid input.
- **Misuse Case Testing:**
  - Executing a malicious act against a system, attackers won't do what normal users would, we need to test misuse to ensure our application or software is safe.
- **Test Coverage Analysis:**
  - Identifies how much of the code was tested in relation to the entire application.
  - To ensure there are no significant gaps where a lack of testing could allow for bugs or security issues to be present that otherwise should have been discovered.
  - With 50+ millions line of code in a Windows OS, often spot checks on critical areas are only enforced.

- Now that we have completed our tests, just like on our log reviews, we need to use it and analyze the data we got from the testing.
- It can be huge amounts of data, and we need to prioritize what we act on first, what is acceptable and what is not.
- Think of the qualitative risk analysis, if it is low likelihood and low impact we may leave it alone and focus on higher priority items.

- **What we covered in the Sixth CBK Domain:**
  - In this domain we covered how we test the actual security levels of our organization, including vulnerability scanning, penetration testing, security assessments, and audits.
    - The **Vulnerability Scanning** can show us half of the Risk = Threat × Vulnerability equation.
    - In the **Penetration Testing** we try to match those vulnerabilities with threats, so we can show the actual risk, if there is no real threats to a vulnerability, we may focus on vulnerabilities with actual threats.
    - How we perform both **Internal and External Audits** and **Full Security Assessments.**
    - Finally how we **review and test the code**, the different methodologies and approaches we use to test different elements of the code.